Query Transformations for Result Merging

Shriphani Palakodety
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
spalakod@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
callan@cmu.edu

ABSTRACT

This paper describes Carnegie Mellon University's entry at the TREC 2014 Federated Web Search track (FedWeb14). Federated search pipelines typically have two components: (i) resource-selection, and (ii) result-merging. This work documents experiments to modify queries to merge results in the federated-search pipeline. Approaches from previous attempts at solving this problem involved custom query-document similarity scores or rank-combination methods. In this document, we explore how term-dependence models and query expansion strategies influence result-merging.

Categories and Subject Descriptors

H.4 [**Information Search and Retrieval**]: Federated Search

General Terms

Algorithms, Experimentation

Keywords

Federated search, result merging, distributional word vectors, term dependence, query expansion

1. INTRODUCTION

Federated search deals with the problem of aggregating results from multiple search engines. The invidual search engines are (i) typically focused on a particular domain or a particular corpus, (ii) employ diverse retrieval models, and (iii) do not necessarily expose statistics used in information retrieval algorithms.

The problem of federated search thus involves (i) analyzing a query to determine which search engines are appropriate for addressing the information need (*resource selection*), and (ii) merging the results returned by each of these engines (*result merging*).

The TREC Federated Web Search Track is a setting for evaluating approaches to federated search. The FedWeb14 track contained three components: (i) vertical selection, (ii) resource selection, (iii) results merging. Vertical selection involves predicting the quality of verticals (like sports and news) for a query. Resource selection involves ranking the available search engines given a particular query. Result merging involves mixing results from a few chosen resources for a given query. A typical system usually leverages vertical selection and resource selection for producing a ranked list of resources (search engines). Then, the query is issued to a few highly-ranked resources and the documents returned by these resources are merged in the result-merging phase.

In this work, we focus on the result-merging phase of federated-search systems. In particular, we explore some techniques that either modify or expand the query-terms to improve performance on result merging tasks. Among the approaches implemented, we leverage term-dependence models and neural network word embeddings.

In the following sections, we describe existing approaches, the methods implemented in this work and an evaluation of the methods.

2. RELATED WORK

Federated search is a well-explored problem in information retrieval research. The subproblems of resource-selection and result-merging have been well studied in the past. Shokouhi & Si [16] presented a comprehensive survey of techniques in federated search. Si & Callan [18] presented a semi-supervised approach to result merging in that used the documents acquired by query-based sampling as training data and linear regression to learn the resource and query-specific merging models. Shokouhi & Zobel [17] presented a technique for using documents sampled from a resource for estimating the global scores of documents for a query.

A set of more than 150 real world search engines and query-based samples from each were provided in the TREC FedWeb13, 2013. Several approaches were employed for merging results. For instance, Mourao et al [13] presented approaches that combined several rank-combination techniques. Di Buccio et al [6] presented a round-robin approach for merging results. At TREC 2013, Guang et al [7], Bellogin et al [1] and Pal et al [14] showed approaches where global document scores for ranking were produced using a combination of query-document similarity measures that at times included scores assigned to resources. However, several of the successful methods assumed that (i) the entire set of documents retrieved from the selected resources were available during the result-merging phase, and (ii) documents retrieved by the resources were available for indexing and

searching. In a typical setting, these assumptions are not necessarily valid.

Word embeddings are mappings from a word to a vector which typically belongs to a continuous vector space. These embeddings allow us to reason about the syntactic and semantic words through linear algebra operations like similarity and distance functions. Word-embeddings have been extensively studied in the past. One of the earliest works in this area was the LSI algorithm by Deerwester et al [4]. The LSI algorithm constructed word embeddings from a Term $\times$ Document matrix using a dimension-reduction operation. Recent approaches for constructing these embeddings have leveraged neural networks extensively. Bengio et al [2, 3] demonstrated the benefits of using these embeddings in language modeling. Embeddings produced by neural networks have provided significant gains over the state-of-the-art approaches in several natural language processing (NLP) applications like sentiment classification, word clustering and so on. Mikolov et al [11, 12]) produced word embeddings that have been used in other NLP tasks. Our paper leverages the embeddings from Mikolov et al [12] to augment queries with additional terms and for weighting.

Metzler & Croft [9] modeled the dependencies between query-terms in a query and demonstrated that these queries produced performed better than a query that used individual query terms. In this paper, we use the sequential dependence model (SDM) for modeling term dependencies.

Term weighting approaches have been used extensively in information retrieval systems. Robertson & Zaragoza [15] provide a survey of probablistic models - a few of which contain term-weighting schemes (like the BM25 model). In this paper, we use the word-embedding from [12] for weighing terms.

3. APPROACH

In a federated search pipeline, the result-merging task follows a resource-selection step. The resource-selection step returns a ranked-list of resources. The query (or a transformation) is issued to a few of the top-ranked resources and the results from all these resources are combined in the resource-selection phase.

In FedWeb14, only the snippets returned by each of the resources were provided. Thus, the following approaches operate on an index of all the snippets returned by the top resources for a query. In the following subsections we describe the various methods implemented.

3.1 Unstructured Queries

Each of the documents (whose snippets we have access to) is ranked using a classic retrieval model - Language-Modeling with Dirichlet smoothing [19].

3.2 Sequential Dependence Model Queries

Sequential-dependence models assume a dependence between neighboring query terms. Essentially, the similarity between a query and a document is measured as a weighted combination of (i) a unigram score (each term individually), (ii) an exact-match bigram score, and (iii) an unordered-window bigram score. Table 1 shows an example of a query and its sequential dependence variant.

In this approach, for each query, the new rankings of all the snippets are given by the scores obtained from executing the sequential-dependence query on the index. The retrieval model employed is the standard Indri retrieval model [19].

Query
`burning man tickets`
SDM Query Indri Expression
`#weight(`
λ_1 `#combine(burning man tickets)`
λ_2 `#combine( #1(burning man) #1(man tickets))`
λ_3 `#combine( #uw8(burning man) #uw8(man tickets)))`

Table 1: An example of a query and the associated indri expression for the sequential dependency model (SDM) query.

3.3 Expanding Using Word Embeddings

Word-embeddings are a mapping from words to a vector space. These embeddings often capture and/or preserve linguistic properties of words. This allows scores or probabilities that are computed for a term to be applied to a semantically similar term. A brief introduction to continuous vector representations is provided below and the query-expansion strategies used are described after.

Bengio et al [2] proposed the use of continuous representations of words for language modeling. The intuition behind this approach was that these embeddings could capture semantic similarities between words and thus help overcome data sparsity issues in language modeling tasks. For example, if the sentence `the cat is walking in the room` is observed in the training corpus, then the evidence gathered from this example must generalize to a sentence like `the dog is walking in the house`. Data-sparsity issues can lead to the latter sentence having zero evidence.

Generating a continuous vector representation for each word allows us to transfer evidence from the term `cat` to the term `dog` and from `room` to `house`. The representations for (semantically) similar terms are thus expected to be similar. Several approaches have been studied to construct such representations. Neural network based language models aim to learn these representations and a statistical language model for the underlying text. These models mainly belong to two categories described below.

- Models that learn the word representations and the language model jointly. The language model described in [2] falls in this category.
- Models that learn the word vector representations first and then train the language model with the word vectors. These models are computationally easier to construct.

The Continuous Bag-of-Words model and the Continuous Skip-gram model proposed by Mikolov et al in [10], belong to the latter category. Word vector representations on a Google News corpus with 100 billion words for a vocabulary of 3 million words can be learned in less than one day using modest hardware. Word vectors learned by both models have performed well in several semantic related task evaluations as shown in [10]. An example is shown in Table 2. In this example, the top 5 words close to the word `france` are displayed. It is clear that the retrieved words are semantically similar (at least for this example).

Word	Cosine Similarity
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323

Table 2: Five words most similar to the word france.

3.4 Query Expansion Strategies

We used two approaches to augment a query with additional terms. These approaches find additional terms that are either (i) similar to the query as a whole, or (ii) similar to individual terms in a query. In both approaches, the terms retrieved for a query are similar to a vector (which represents terms or a query aggregate). For computing a vector that represents the entire query, we obtain the vectors for each of the terms and compute the mean vector.

Thus, in the first expansion strategy, we add a few terms to the query that are closest to the query mean vector. In the second strategy, for each term, we retrieve additional words closest to the vector. In both cases, the terms added are obtained from the global vocabulary of the word embeddings available.

Once the additional terms are added to the query, the snippets are scored based on this newer query using the language-model with dirichlet smoothing retrieval model.

3.5 Term Weighting Strategies

Our approach uses word-embeddings from [10] to produce weights for individual query-terms. We use two strategies to weigh terms. In both cases, the weights applied to each term are the distance between the term's embedding and a certain global vector. The distance metric is euclidean distance in both approaches. The intuition behind using distance from a vector is that the farther a term is from the global vector, the more information it contains and thus it merits a higher weight.

In the first of these approaches, the global vector used is the query mean vector obtained by averaging the vectors corresponding to the terms in the query. In the second approach, the global vector used is the average vector of the entire vocabulary of the learned embeddings (close to 3 million words and phrases).

4. EVALUATION

In this section, we elaborate on the FedWeb13 and FedWeb14 collections and present the results.

4.1 Data collections

FedWeb13 contains results sampled from from 157 real world search engines in 24 verticals. 2000 queries were issued to the search engines during the sampling phase. See Table 3 for a summary of data statistics of FedWeb13. In this paper, we report experiments and analysis on the FedWeb13 dataset since as of this paper, the tools for evaluating on FedWeb14 have not yet been released.

The data collection of FedWeb14 is built from 149 web search engines crawled between April and May 2014. 4000 queries were issued to the search engines in the sampling phase. Table 4 presents the data statistics of FedWeb14. For the result merging phase, only snippets were provided.

		Total	Per Engine
Samples (2000 Queries)	Snippets	1,973,591	12,570.6
	Pages	1,894,463	12,066.6
Topics	Snippets	143,298	912.7
	Pages	136,103	866.9

Table 3: FedWeb13 collection statistics.

		Total	Per Engine
Samples (4000 Queries)	Snippets	1,422,758	9548.7
	Pages	3,471,773	23300.5
Topics	Snippets	51458	345.3
	Pages	0	0

Table 4: FedWeb13 collection statistics.

Only the results provided by the organizers are provided for the FedWeb14 dataset since tools for performing a per-query analysis are not yet released (as of this paper).

4.2 Experimental setup

We use the Indri search engine to index and search the snippets for each search engine. Stop-word removal and stemming did not aid system performance significantly and since the embeddings were built on a large english corpus, the risk of missing term vectors is minimal. In case, an out-of-vocabulary term (OOV) was encountered, we did not include the term vectors. All our approaches used a classic retrieval model - language model with dirichlet smoothing. The parameter μ for this retrieval model was set the Indri default of 2500. For the sequential-dependence model implementation, the weights assigned to the unigram, exact-match bigram and window bigram components were 0.5, 0.25 and 0.25 respectively. For the query-expansion strategies, at most 5 additional terms with cosine similarity scores above 0.7 were chosen for both the strategies. For term weighting, OOV terms were assigned a default weight of 1.0. The word-vector representations used were 300-dimensional vectors released by Google, trained on a Google News corpus of about 100 billion words.

We assessed the runs with the `gdeval.pl` tool provided by TREC and focus on NDCG@20 for the result merging task. The results for the FedWeb13 data are in Table 5. Performance of our system alone on FedWeb14 is provided in Table 6 (since the best runs were not available at the time of submission). In both tables, `plain` refers to the approach in section 3.1, `sdm` refers to the approach in section 3.2, and `Exp-Avg` and `Exp-Term` refer to the query-expansion strategies explained above.

All the result-merging scores were based on baseline resource-selection runs provided by the organizers. In addition to the best performing system we include the best performing system that only used snippets since the FedWeb13 task allowed participants to use the documents returned by each of the resources during the result-merging phase. In FedWeb14, only snippets were available for use. The results of the plain retrieval model, the SDM queries and the expansion strategies are also provided for FedWeb14. The term-weighting approach was not submitted to the FedWeb14 task and thus we only provide results on FedWeb13 for this approach.

Table 5 lists the performance of our system and the best FedWeb13 runs. We observe that (i) in all cases, using only snippets as opposed to documents automatically leads to a

Approach	NDGC@20
FedWeb13-Docs-Best	0.257
FedWeb13-Docs-Median	0.162
FedWeb13-Snippets-Best	0.161
FedWeb13-Snippets-Median	0.142
FedWeb13-plain	0.210
FedWeb13-sdm	0.224
FedWeb13-expansion-1	0.188
FedWeb13-expansion-2	0.201
FedWeb13-weighting-1	0.213
FedWeb13-weighting-2	0.211

Table 5: FedWeb13 collection statistics.

Approach	NDGC@20
FedWeb14-Best	0.323
FedWeb14-Median	0.289
FedWeb14-plain	0.277
FedWeb14-sdm	0.276
FedWeb14-expansion-1	0.285
FedWeb14-expansion-2	0.286

Table 6: FedWeb13 collection statistics.

massive drop in system performance. This is consistent with the observations of the FedWeb13 organizers [5]. Thus, for a realistic comparison we only consider the best submission from FedWeb13 that did not use documents (shown as `FW13-SNIPPET-BEST`). Our approaches clearly outperform the best result-merging score from the FedWeb13 track (that only considered snippets). In particular we note that most of the models perform very similar to each other and there is a minor performance drop when expanding a query with terms close to the query mean vector.

We also report results for some of our techniques on the FedWeb14 corpus (shown in Table 6). In this corpus, we notice that our approaches are extremely close to the median score and the performance gap between our approach and the best system is slightly larger than the gap for the FedWeb13 corpus. In this case, the expansion strategies slightly outperform the other approaches.

On a per-query basis, there are no particular kind of queries in the FedWeb13 corpus that were aided by our approaches. Between the various approaches implemented, the variance is not particularly high.

5. CONCLUSIONS

In this work, we explored how query transformations can be leveraged for merging results in the federated search pipeline. The first observation from the FedWeb13 collection is that when restricted to using snippets, the performance drops quite severely - an observation made by the organizers as well in [5]. The best performance on the FedWeb13 dataset was obtained by employing sequential-dependence models. The query-expansion approaches did not provide a performance improvement compared to sequential-dependence models and classic retrieval models like the language-model with dirichlet smoothing. On FedWeb14 however, the query-expansion using word-vector provided a slight improvement in performance. Newer advances in learning continuous representations of paragraphs or documents (as demonstrated by [8]) can be leveraged in the future to provide a more principled approach to query expansion and document(snippet) representation.

6. ACKNOWLEDGMENTS

This research was in part supported by National Science Foundation (NSF) grant IIS-1160862. Any opinions, findings, conclusions, and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] A. Bellogín, G. G. Gebremeskel, J. He, J. Lin, A. Said, T. Samar, A. P. de Vries, and J. B. Vuurens. CWI and TU delft at TREC 2013: Contextual Suggestion, Federated Web Search, KBA, and Web Tracks. In *The 22nd Text Retrieval Conference (TREC 2013)*, 2013.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003.

[3] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[4] S. C. Deerwester, S. T. Dumais, and R. A. Harshman. Indexing by latent semantic analysis. 1990.

[5] T. Demeester, D. Trieschnigg, D. Nguyen, and D. Hiemstra. Overview of the TREC 2013 Federated Web Search Track. In *The 22nd Text Retrieval Conference (TREC 2013)*. TREC, 2013.

[6] E. Di Buccio, I. Masiero, and M. Melucci. University of Padua at TREC 2013: Federated Web Search Track. 2013.

[7] F. Guan, Y. Xue, X. Yu, Y. Liu, and X. Cheng. ICTNET at Federated Web Search Track 2013. In *The 22nd Text Retrieval Conference (TREC 2013)*, 2013.

[8] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *Proceedings of The 31st International Conference on Machine Learning*, 2014.

[9] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, pages –1–1, 2013.

[11] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[13] A. Mourao, F. Martins, and J. Magalhaes. NovaSearch at TREC 2013 Federated Web Search Track: Experiments with Rank Fusion. In *The 22nd Text Retrieval Conference (TREC 2013)*, 2013.

[14] D. Pal and M. Mitra. ISI at the TREC 2013 Federated task. In *The 22nd Text Retrieval Conference (TREC 2013)*, 2013.

[15] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, Apr. 2009.

[16] M. Shokouhi and L. Si. Federated search. *Foundations and TrendsÂő in Information Retrieval*, 5(1):1–102, 2011.

[17] M. Shokouhi and J. Zobel. Robust result merging using sample-based score estimates. *ACM Trans. Inf. Syst.*, 27(3):14:1–14:29, May 2009.

[18] L. Si and J. Callan. A semisupervised learning method to merge search engine results. *ACM Trans. Inf. Syst.*, 21(4):457–491, Oct. 2003.

[19] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, Apr. 2004.